Final Exam

```
    ____ _____ ____ ____    ________ __
   / ____/ ____/ ___/ ___/  /__ /__   / _ \
  /  _/ /  _// /    \___ \    /_ \  / / / / /
 / /___/ /__/ /___ ___) /  ___) |/ /| |_| |
/_____/_____\____/____/  /____//_/  \___/
```

EECS 370 Winter 2023: Introduction to Computer Organization

You are to abide by the University of Michigan College of Engineering Honor Code. Please sign below to signify that you have kept the honor code pledge:

I have neither given nor received aid on this exam, nor have I concealed any violations of the Honor Code.

Signature: ______________________________

Name: ______________________________

Uniqname: ______________________________

First/Last name of person sitting to your ***Right***
(Write ⊥ if you are at the end of the row) ______________________________

First/Last name of person sitting to your ***Left***
(Write ⊥ if you are at the end of the row) ______________________________

Exam Directions:

- You have **120 minutes** to complete the exam. There are **9** questions in the exam on **6** pages (so 12 "sides" total). **Please flip through your exam to ensure you have all 6 pages.**
- You must show your work to be eligible for partial credit.
- Write legibly and dark enough for the scanners to read your answers.
- **Write your uniqname on the line provided at the top of each page. Do this at the beginning of the exam; you will NOT be given time to do it at the end.**

Exam Materials:

- You are allotted **one 8.5 x 11 double-sided** note sheet to bring into the exam room.
- You are allowed to use calculators that do not have an internet connection. All electronic devices with an internet connection are strictly forbidden.

uniqname: ____________

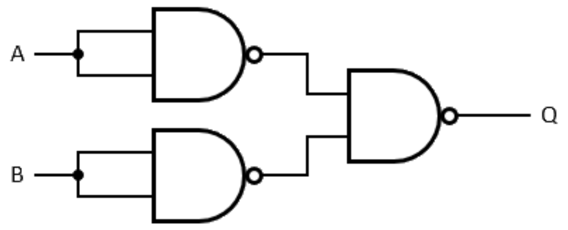
1. Multiple choice [15 pts]

Completely fill in the circle of the *best* answer.

1. Under what conditions would you expect a write-through cache to have a lower number of bytes transferred between the cache and memory than a write-back cache? **[2]**
   - ○ The program has low spatial locality but high temporal locality
   - ○ The program has high spatial locality but low temporal locality
   - ○ The program has high spatial locality and high temporal locality
   - [✓ The program has low spatial locality and low temporal locality]
   - ○ Never

2. Functions are surprisingly difficult for the branch predictors we've discussed to deal with. What is it about functions that typically cause problems for those predictors? **[2]**
   - ○ It is often hard to predict the "direction" of a function call.
   - ○ It is often hard to predict the "direction" of the return from a function.
   - [✓ It is often hard to predict the target of the return from a function.]
   - [✓ It is often hard to predict the target of a function call.]
   - ○ The branches associated with function calls and returns have very little spatial locality.

3. Which of the following formulas is equivalent to the circuit below? **[2]**



$(A \cdot A)' = A'$

$(B \cdot B)' = B'$

$(A' \cdot B')'$

$A + B$

   - ○ Q = A nor B
   - ○ Q = A and B
   - [✓ Q = A or B]
   - ○ Q= not(A) or not(B)
   - ○ None of the above

4. When comparing direct-mapped caches to fully-associative caches that otherwise have identical parameters, which of the following would be expected to be true? **[3]**
   a) Direct-mapped caches will have a lower hit latency, fully-associative caches will have a higher hit rate ✓
   b) Direct-mapped caches will require more index bits, fully-associative caches will have more tag bits. ✓
   c) Direct-mapped caches will require fewer block offset bits, fully-associative caches will have more LRU bits. ✗
   - ○ Only a
   - ○ Only b
   - ○ Only c
   - [✓ Only a and ~~b~~]
   - ○ Only b and c
   - ○ Only a and c
   - ○ All of a, b, and c.

For the next two questions, assume you have a byte-addressable, 256-byte virtually addressed cache with 16-byte blocks. Assume all entries in the cache start as "invalid" and addresses are 16-bits. All loads and stores are to 4-byte values. [illegible]

5. For which of the following access patterns will a direct-mapped cache will get a better hit-rate than a two-way associative cache using LRU replacement? **[3]**
   - ○ 0x0000, 0x0010, 0x0020, 0x0000
   - ○ 0x0000, 0x0080, 0x0080, 0x0001
   - [✓ 0x0000, 0x0180, 0x0080, 0x0000] ✓
   - ○ 0x0000, 0x0060, 0x0080, 0x0001
   - ○ None of the above

   000 101 000 000.

$256 \div 16 = 2^8 \div 2^4 = 2^4$.

8 set

tag	block offset
	4.

tag.	set index	block offset
	3.	4

000 —
001
002

set0 00 0
set1 00 1
set2 00 2

6. For which of the following access patterns will a fully-associative cache using LRU replacement will get a better hit-rate than a two-way associative cache? **[3]**
   - [✓ 0x0000, 0x0100, 0x0200, 0x0000]
   - ○ 0x0000, 0x0001, 0x0200, 0x0002
   - ○ 0x0000, 0x0010, 0x0020, 0x0000
   - ○ 0x0000, 0x0410, 0x0020, 0x0000
   - ○ None of the above

000
008 2 hit.

000 ✓
010
020

000 tag 00 0
000 tag 01 0
000 tag 02 0
000 tag. 00 0.

	set index	tag.
0x0000	000	0x00 0
	set index.	
0x0080	000	0x00 1
0x0000	000	0x00 0
0x0180	000	0x01 1
0x0080	000	0x00 1
0x000?		

4:00-4:06 500xMHz

2. **True or False** 0.5×10^9 **[13 pts]**

Complete the following true or false questions.

(1) A clock with a 2ns period has a frequency of 200MHz. $\frac{1}{2\times10^{-9}}$ 2 True [✓ False]

(2) The number of LRU bits required for a set associative cache depends on cache associativity. [✓ True] False

(3) An XOR gate can be created using only AND gates. True [✓ False]

(4) A multi-level page table can take up more memory space than a single level page table. [✓ True] False

(5) In the 3C's cache model, a "compulsory" cache miss can sometimes be avoided by changing the cache's total size while holding block size constant. True [✓ False]

(6) A virtually-addressed cache doesn't need to access the TLB to see if the cache will get a hit or a miss. [✓ True] ~~False~~

(7) Virtual address space is generally limited to the amount of DRAM on a computer. True [✓ False]

(8) Dennard scaling is the claim that each transistor will generally use the same amount of power no matter how small the transistor is. True [✓ False]

(9) The size of a virtual page can be larger than the physical page. True [✓ False]

(10) A 2-bit branch predictor can get about a 0% hit rate on a branch that alternates between taken and not-taken forever (so T, N, T, N, T, N...) [✓ True] False

01 10 01 10
N T N T

(11) In the 3 C's model, you would expect to be able to reduce the number of conflict misses by increasing the associativity of the cache. [✓ True] False

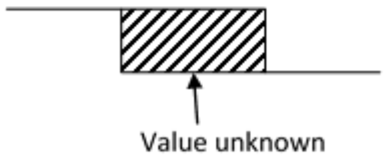
(12) Tags in TLBs are derived from virtual page numbers. [✓ True] False

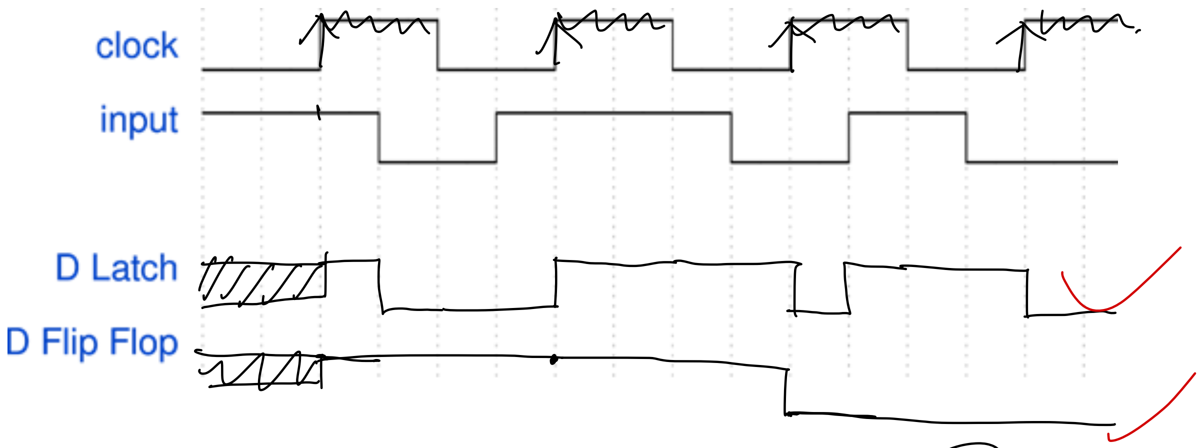
(13) If you increase the page size while holding DRAM's size constant, you would expect the number of physical pages to increase. ↓ True [✓ False]

3. Older Stuff [9 pts]

Things from days of yore.

1. Complete the timing diagram below for both a D latch and a rising-edge triggered D flip-flop. If a value is unknown, indicate that clearly using the notation shown. Assume there is no meaningful delay. **[4]**





2. Using 2's complement notation, write the *8-bit binary* representation of -22. **[2]**

0b 1 1 1 0 1 0 1 0

-22 $\frac{1}{-32}$ $\frac{0}{16}$ $\frac{1}{8}$ $\frac{0}{4}$ $\frac{1}{2}$ $\frac{0}{1}$

3. Consider an 8-bit floating point format based on the IEEE standard where the most significant bit is used for the sign (as the IEEE format), the next 3 bits are used for the exponent and the last 4 bits are used for the mantissa. The scheme uses "biased 3" to represent the exponent (rather than biased 127 used for a 32-bit IEEE floating point number) and has an implicit one just like the IEEE format. This scheme is called "VSF" (very short float).

Write the *binary* encoding of -2.5 as a VSF number. **[3]**

1 $\underbrace{1\ 0\ 1}_{\text{exponent.}}$ 0 1 0 0

0b 1 1 0 1 0 1 0 0 .

2.5 → 10 . 10

$\boxed{10}.10$

$1.\boxed{010} \times 2^{\boxed{1}}$

↓ mantissa.

↗ exponent.

0 → 011

2 → 5

0 1 2

4 5.

101

100

$\frac{1}{4}$ $\frac{0}{2}$ $\frac{1}{1}$

uniqname: ____________

4. Pipeline stalls and forwarding [8 pts]

Determine data hazards and avoidance methods in a pipeline

Consider a **5-stage** LC2K pipeline datapath that uses **detect-and-forward** to resolve data hazards, **detect-and-stall** to resolve control hazards (no branch prediction), and has **internal forwarding** for its register file.

Determine the number of pipeline stalls for each of the following benchmarks. Also, specify *all* (could be more than one) the instructions that *receive* forwarded data by shading the circles. **Ignore and do NOT specify instructions that received data through register internal forwarding.**

Benchmark 1:

```
    beq    5   6    end    //Not Taken   3 noop.
○   lw     0  [1]   data1
○   lw     0  [1]   data2
○   nor    3   4   [7]
●   add    2   1    3
```

# of Stalls : 3

Benchmark 2:

```
    add    2   2   [3]
○   lw     0  [2]   str    < 1 noop.
●   lw     2  [7]   data3  < 1 noop.
●   beq    1   7    if     //Not Taken   3 noop.
○   add    7   5    1
```

# of Stalls : 5

Benchmark 3:

```
    add    3   3   [2]
○   nor    4   5   [6]
○   lw     0  [1]   data4  < 1 noop
○   sw     0   1    data5
○   add    1   1   [2]
```

# of Stalls : ~~0~~ 1

Benchmark 4:

```
    nor    4   5   [6]
○   add    1   2   [3]
○   lw     1  [2]   data6
●   add    2   2   [4]    < 1 noop.
○   nor    2   2   [3]
```

# of Stalls : 1

uniqname: ____________

5.	**Pipeline Performance** **[15 pts]**
	Perform performance calculations on a given pipeline with a cache

Consider the following 5-stage LC2K pipeline:

- **Detect-and-forward** is used to handle data hazards.
- **Speculate-and-squash** is used to handle control hazards.
- When a lw or sw instruction accesses the memory system in the MEM stage, either
  - There is a **cache hit** (95% of time time) and the pipeline **does not stall**
  - or, there is a **cache miss**, which causes the pipeline to **stall for 20 cycles** while the main memory is accessed.
- 1% of instruction fetches from an instruction cache are cache misses and result in a stall of 20 cycles.
- *Throughout this problem* you are to assume that no sources of stalls will overlap.

Say we run a program with the following characteristics:

lw	30%
sw	10%
add/nor	40%
beq	20%

- 25% of each type of instruction that writes to a register (**lw, add, nor**) is immediately followed by an instruction that depends on it.
- 5% of instructions that write to a register (**lw, add, nor**) are immediately followed by an independent instruction, and then immediately followed by a dependent instruction.
- 35% of branches are mispredicted.

1) Complete the equation for CPI below using data given above. It is fine to leave your answer as an equation that can be plugged into a calculator. **[3]**

CPI = 1

+ $0.2 \times 0.35 \times 3$ (increase due to control hazards)

$0.3 \times 0.25 \times 1$

+ ____________ (increase due to data hazards)

$1 \times 0.01 \times 20 + (0.3 + 0.1) \times 0.05 \times 20.$

+ ____________ (increase due to cache misses)

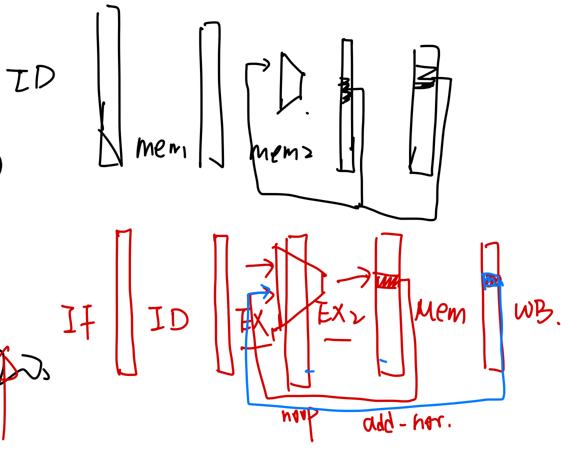
2) Say with a process shrink (i.e. the transistors are made smaller) we increase the clock frequency of the processor by a factor of 2 (including the cache but not the memory). In order to make this work, we had to split the execution stage into two stages (EX1 and EX2, where all ALU operations finish in EX2). Branches still resolve in the MEM stage. If we run the same program from part (a) on our new pipeline, what is the new CPI? It is fine to leave your answer as an equation that can be plugged into a calculator. **[9]**

CPI = 1

+ $0.2 \times 0.35 \times 4$ (increase due to control hazards)

+ $0.3 \times 0.25 \times 1$ (increase due to data hazards)

+ $1 \times 0.01 \times 400 + (0.3 + 0.1) \times 0.05 \times 400$ (increase due to cache misses)



a) Say for part 1) you had found a CPI of 2.0 and for part 2) you had found a CPI of 3.0. What would be the speedup[1] after our process shrink expressed as a percentage? It is fine to leave your answer as an equation that can be plugged into a calculator. **[3]**

1.33 %

$2.0 \times 2 = 4$

$3.0 \times 1 = 3$

$\frac{\frac{1}{3}}{\frac{1}{4}}$

$A = 4$

$4 - 3 = 1$

$B = 3.$

$3\overline{)4}$ 1.3, 3, 10, 9, 1

$\frac{\frac{1}{3}}{\frac{1}{4}} = \frac{1}{3} \times 4 = \frac{4}{3}$

$\frac{\frac{1}{2}}{\frac{1}{1}} = 50\%$

$\frac{\frac{1}{1}}{\frac{1}{3}} = 3$

2 1

[1]Recall that in general, if we say Processor A is 50% of the speed of processor B, we mean A is half as fast. Which is the same as saying that A takes twice as long to do the task. If we say Processor A is 300% the speed of processor B, that means it is 3 times as fast.

↑ 1 $\frac{A}{B}$ 3

6.	**Cache Basics** **[8 pts]**
	Just the facts

Indicate the number of bits used for the index and block offset of each of the following caches. Assume the address size (physical and virtual) is 32-bits and that memory is byte addressable.

1) A 128KB, 4-way associative cache with 32-byte blocks. Index: 10 Offset: 5

$128KB \div 32B = 4K$ 2^5

2^{12} line 4-way

K set

2) A 1MB, direct-mapped cache with 16-byte blocks. Index: 16 Offset: 4

$2^{20}B$ 2^4

$2^{20}B \div 2^4 = 2^{16}$

3) A 48KB, 3-way associative cache with 8-byte blocks. Index: 11 Offset: 3

$48KB \div 8B = 6K$ line. 2^4

set $\boxed{E3}$

2K set. set index : 2×2^{10} 2^{11}

4) A 1MB fully-associative cache with 128-byte blocks. Index: 0 Offset: 7

2^{20} 2^7

$4 \times 4 \times 4 \times 4 \times 2.$

$16 \times 8 = 1$[illegible]

page 16-byte

16byte cache
2byte block.

uniqname: ________

8 line. 2-way 4 set.

7. Examining the Memory System Bit by Bit [10 pts]

Working with the data

Consider a byte-addressable architecture with 12-bit virtual addresses. The system has a maximum of 1KB of physical memory with 16-byte page sizes. The system has a 16-byte 2-way set associative physically-addressed cache with a 2-byte block size and a fully-associative TLB with 4 entries. The TLB, cache contents, and a *snippet* of the single level page table are provided below.

12 bit

(tag | set bit | offset |
9 bit 2 bit 1 bit.

Page table

VPN	PPN	Valid
0x00	0x01	1
0x01	0x03	0
0x02	0x0F	1
0x03	0x00	1
0x04	0x22	1
0x05	0x1A	0
0x06	0x31	0
0x07	0x13	1

VPN	PPN	Valid
0x08	0x25	1
0x09	0x26	1
0x0A	0x3A	1
0x0B	0x3A	0
0x0C	0x1B	1
0x0D	0x1C	1
0x0E	0x27	1
0x0F	0x1F	0

TLB

Tag	PPN	Valid
0x02	0x0F	1
0x2B	0x1A	1
0x06	0x0A	0
0x0D	0x1C	1

physicall

$\boxed{0x27B}$

Cache

Set Index	Tag	Valid	Byte0	Byte 1
0	0x37	1	0xDE	0xAD
	0x1A	0	0x12	0xB0
1	0x65	1	0x0A	0xC1
	0x4F	1	0x99	0x1F
2	0x1C	1	0x84	0x92
	0x00	1	0xBE	0xCF
3	0x7B	1	0xCC	0xA0
	0x0A	1	0x45	0x67

Say that the processor reads one byte from virtual address **0x0EB**. Recall that the cache is physically addressed. Answer the following questions. Provide all numeric answers in hex. If the answer is unknown, write "unknown". Note that early errors on this problem could cause later answers to also be wrong.

B: 11

$\frac{1}{8}\ \frac{0}{4}\ \frac{1}{2}\ \frac{1}{1}$

1) In hex, what is the virtual page number associated with that address? **[2]** 0x 0E ✓

2) Is this a TLB hit? **[1]** Yes [✓ No] ✓

3) In hex, what is the physical page number associated with this access? **[2]** 0x 27 ✓

4) What set index could hold the data? **[2]** 0x 1 ✓

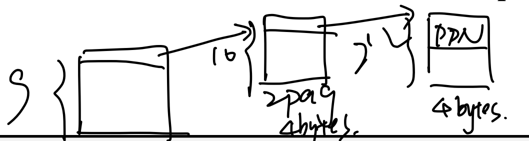
5) What is the value of that byte of memory being accessed? **[3]** 0x ~~unknown~~ 0x1F

0x27B.
4 7

0 0 1 0 0 | 1 1 1 1

8 4 2 1 8 | 4 2 1

$\frac{1}{8}$ **uniqname:** ______________



physical memory. 32GB.

8. Hierarchical Page tables [10 pts]

1 pag 2^{11} B $2\times2^{11}=2^{12}$ B $2^{12}\div4=2^{10}$ B $2^5\times2^{30}$

Clearly write answers in the blanks provided. 9 $10+12+11=22+12=34$ $42-34=8$. 2^{35}

Virtual 42 $2^{12}\div4=2^{10}$.

Consider a 42-bit byte-addressable system that supports virtual memory with the following specifications:

virtual | level1 | level2 | level3 | page offset | — 9, 10, 12, 11 bit

- A page is 2 KB.
- The page table is hierarchical with 3 levels.
- The first-level page table occupies exactly 1 page of memory.
- Each second-level page table occupies exactly 2 pages of memory.
- All page table entries are 4 bytes .
- A maximum of 32 GB of physical memory can be installed.

page 2KB 2^{11}B $2^{12}\times4$ $\div2^{14}$ 2^{11}

| PPN 24 bit | page offset 11 bit |

Provide a number (rather than an equation) in each blank for parts 1-6. Something like 2^{20} is fine. $2^{22}/4$ is not.

8 2^3 8

$35-11=24$

1. How many bits are used for the page offset? **[1]** __11__ bits

2. How many virtual pages exist in this system? **[1]** $2^{42}\div2^{11}=2^{31}$ __2^{31}__ pages

3. How many physical pages can exist in the system? **[1]** __2^{24}__ pages

$32GB\div2^{11}B=2^{35}\div2^{11}=2^{24}$

4. How many bits in a virtual address are used to index the first-level page table? **[1]**

__9__ bits

5. How many bits in a virtual address are used to index a second-level page table? **[1]**

__10__ bits

6. How many bits in a virtual address are used to index a third-level page table? **[2]**

__12__ bits

7. In the worst case, how many pages would this page table occupy? **[3]**

__$1+2^9 + 2^{22}$__ pages (You can write an "equation", rather than number, for this one)

2^{10}

$1+2^9$ page $\times2$ $+2^9\times2^{10}\times8$ page

$1+2^9\times2+2^{19}\times8$ 2^{10}

$1+2^9+2^{22}$

1024B ÷ 32B line index 5 bit.

$2^{10} \div 2^5 = 2^5$ line.

uniqname: ____________

tag	line index	offset
	5	5

9. Cache Analysis [12 pts]

Deeper thoughts about caches

1) Consider a 1024-byte **direct-mapped cache** with a block size of 32 bytes. The cache starts empty and P, Q, and R are addresses. You are given the following stream of address references. Cache misses are marked as "M", while hits are marked as "H".

P.Q not the same.

~~QR~~ ~~R~~ ~~QR~~ P

Address	P	Q	R	P	Q
Cache Access	M	M	H	M	M

a. From the above you can be ***sure*** of which of the following? Select *all* that are true. **[3]**

- ○ P and Q are in the same cache block
- ○ P and R are in the same cache block
- ● R and Q are in the same cache block ✓
- ○ You cannot be sure of any of the three above.

b. From the above you can be ***sure*** of which of the following? Select *all* that are true. **[3]**

- ● P and Q have the same line index, but are in different cache blocks
- ● P and R have the same line index, but are in different cache blocks
- ○ R and Q have the same line index, but are in different cache blocks
- ○ You cannot be sure of any of the three above.

2) Consider a 1024-byte **2-way associative cache** with a block size of 32 bytes. The cache starts empty and A, B, C, D, and E are addresses. You are given the following stream of address references. Cache misses are marked as "M", while hits are marked as "H".. 2^5 6.

Address	A	B	C	D	E	D	C	B	A	B
Cache Access	M	M	H	M	M	H	M	H	H	D

A C

A B

a. From the above which of the following ***could*** be true? Select *all* that could be true. **[3]**

- ○ A and B are in the same cache block
- ● A and C are in the same cache block
- ● B and C are in the same cache block
- ○ D and E are in the same cache block

1024 2^{10} $\div 2^5 \div 2 = 2^4$

tag	set index	offset
	4 bit	5 bit

A ~~E~~ ~~BC~~E C BC D→D

b. From the above which of the following ***could*** be true? Select *all* that could be true. **[3]**

- ○ A and B map to the same set, but different cache blocks
- ○ B and C map to the same set, but different cache blocks
- ● B and D map to the same set, but different cache blocks
- ● D and E map to the same set, but different cache blocks

E ~~A C~~ A C

B E

D E

A C

B

D